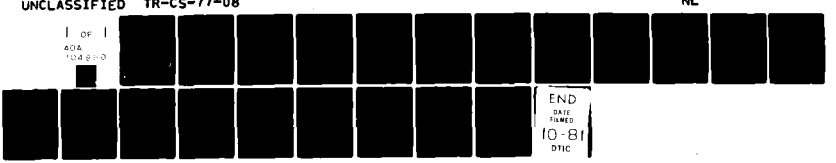
AD-A104 950 KANSAS STATE UNIV MANHATTAN DEPT OF COMPUTER SCIENCE F/G 9/2
DESIGN CONSIDERATIONS IN DISTRIBUTED DATA BASE MANAGEMENT SYSTE--ETC(U)
APR 77 P S FISHER, F J MARYANSKI DAAG29-76-G-0108
UNCLASSIFIED TR-CS-77-08 NL



END
DATE FILMED
10-81
DTIC

(15)

DEPARTMENT

of

COMPUTER SCIENCE

KANSAS STATE UNIVERSITY

(6) Design Considerations in Distributed Data Base Management Systems,[1]

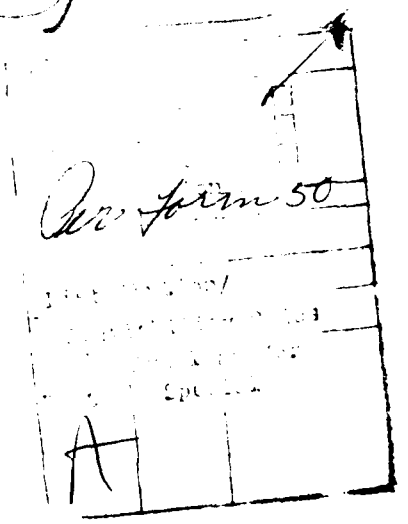
(14) TR-CS-77-08

(11) Apr[redacted]77

(10) Paul S. Fisher
Fred J. Maryanski

(12) 21

Computer Science Department
Kansas State University
Manhattan, Kansas 66506



(15) DAAG29-76-G-0108

[1]The work reported herein was supported by the United States Army Computer Systems Command, Grant No. DAAG-29-76-G108.

(5) 391123

I. INTRODUCTION

As the computer industry has grown in experience and application, there is an area which has received much attention; that is, the area of managing, organizing and storing data. Problems of reliability, security and timeliness with data have drawn almost every computer user's attention. Since the late 60's, there has been a growing interest in providing software which separates (at a cost) the manipulation aspect from usual organizational and accounting functions found in a normal program. Such "housekeeping" chores can require significant efforts on the part of programmers. Further, once such chores are initially done they will have to be redone time and again, since requirements and usage will change.

With the advent of Data Base Management Systems (DBMS) and associated facilities (Data dictionaries, query languages, report writers, etc.) the task of data organization, management, and storage has been given to a select group of specialists. These specialists (the Data Base Administrators (DBA) provide the necessary control, logging, and access information and software to the program. Such activity relieves the programmers of this overhead function allowing them to concentrate on the necessary manipulations.

This paper focuses on some alternatives with respect to a DBMS in terms of a centralized versus decentralized environment. The first section of this paper deals with the concepts and tradeoffs involved in considering the two environments. The second section of the paper then deals with problems which are encountered in a distributed data base management system. These problems include deadlock, rollback and recovery, data conversion, redundancy, and communication and operating system requirements for effective distribution.

II. THE DBMS

If the total data processing application of an organization is a payroll, which is a sequential operation, then there is good reason not to acquire a DBMS. However, if data is accessed randomly, and/or each separate department maintains its own copy and version and any one of a number of other problems exists, then a DBMS may be of some use. At least a new dimension of problems will be introduced, hopefully at the expenses of a far larger number of problems.

DBMS systems support essentially four different structures: Network, Hierarchical, Inverted, and Relational (no fully relational commercial DBMS is available yet). Each of the techniques has its own advantages and disadvantages which are not discussed further here, although careful study is required before acquiring a DBMS.

The DBMS will require from minimal to significant machine resources. It can increase file size requirements, slow processing, require additional manpower resources, and be of a significant cost. However, through its use resources can be released, data more accurately controlled, programs developed more quickly, and the continual modifications and changes incorporated without noticeable effect on the user.

DBMS systems are found on every architecture from the small to the very large. In fact, more than one DBMS is found on the full spectrum of machine architectures with user transparency more real than imagined. Because of this feature it is quite possible now to consider as a reasonable alternative to a single, centralized, large system supporting a DBMS, a spectrum of machines each supporting a local DBMS and each in support of the other.

III. THE ALTERNATIVE - DISTRIBUTED DBMS

With continued growth in data and processing requirements come the

continual requirement for upgrading and increasing storage and processing facilities. As a viable option to growth of the single large processor, the possibility exists that required capabilities can be obtained by addition of smaller processors in the fashion shown in Figures 1.a - 1.c. Each figure portrays an increasingly complex environment. Within the three figures there are three functional machine categories shown, and these are described in the following paragraphs.

The first is that of a host. This machine can be of any size and architecture. Typically it is a large machine which serves the complete variety of users. This machine may also support DBMS activities.

The second machine category shown is that of a back-end machine. This machine can also be any one of a variety of architectures, although typically it is a minicomputer. The purpose of this machine is to support some or all of the data base activity. A typical software configuration for the host and back-end is shown in Figure 2. Essentially, the operation of this configuration is as follows: A host program requiring a DBMS function executes a call to the DBMS monitor. This residual monitor forwards the call to the back-end machine which allocates its resources (if available) to provide the requested service. In order to satisfy the requested service, the back-end may have to execute several disk reads, along with substantial processing. Once the requested information is obtained, it is returned back to the host monitor which makes it available to the requesting task. Figure 3 shows the relationship obtained by simulation between data base requests on a host only, a host and back-end where the back-end provides from zero to five levels of multi-programming [1].

The last machine category shown in Figure 1.c is that of a bi-functional machine. In this situation a machine is both serving in the back-

end mode as well as the host mode. However, unlike the strictly host/back-end function which implied that machines be co-located, bi-functional machines can provide the environment for a distributed data base system.

Before dealing with specific problems attendant in a distributed network, it is worth while to consider some advantages and disadvantages associated with both centralized and decentralized concepts. Neither option is clearly advantageous in all cases, and hence the decision process must be carefully considered.

The advantages of a centralized system are concerned with control, protection, and redundancy. With respect to control, a central facility can provide better qualified DBA personnel to control growth, development and deployment of data. Another advantage is in the area of protection. It is obvious that data collected in one location can be more easily protected than data which is distributed. The last area to be mentioned is that of redundancy. All too often the temptation exists to proliferate copies of data. This proliferation with its attendant problems is best controlled within a central facility.

The disadvantages to a single centralized system is alleviated by the introduction of back-ends or local networks. However, if one considers the constant upgrade problem which can require from slight to major software modifications coupled with the ever increasing complexity of supplied equipment and software, then the problems can be truly overwhelming. Unfortunately, although source language standards exist, they are almost always ignored. The last disadvantage occurs in costs of the ever increasing systems. For this reason, upgrades are often delayed, which causes repercussions in service and growth.

With respect to decentralized systems, the advantages lie in the

areas of data availability, smaller physical machine requirements, and a potential for providing better service. Obviously, if a particular group of users requires access to a collection of data, the data and the machine should be located in proximity to those users. In most cases, the supplied machine resources will be smaller, easier to maintain, service, and use. Furthermore, any required upgrade in facilities can be made on a local basis, not interfering with other nonrelated users.

The disadvantages stem from a lack of central control with attendant growth in redundant data, and the problems of resolving resultant data descrepancies.

With these considerations we then propose the system as shown in Figure 4. Such a system consists of machine groups (clusters) tightly linked together with high-speed communication lines. It is quite possible to talk about the degenerative case where a cluster consists of only a single machine. In general, a cluster might consist of multiple heterogeneous machines, each of which may be bi-functional or dedicated to a specific task. Between each cluster communication can take place over any standard communciation lines using any communication protocol desired.

IV. PROBLEMS IN DISTRIBUTED DATA BASE SYSTEMS

While distributed data base systems offer many potential advantages (as discussed in the preceeding sections), the data base industry has not yet advanced to the point of producing a commercially available package. Several formidable obstacles lie in the path of a generalized, distributed DBMS. A review of these problems is presented here along with a discussion of potential solutions.

IV.1. DEADLOCK

The phenomenum of deadlock has long been an important problem for operating system designers. In general, deadlock occurs when a set of

tasks have blocked each other from execution by requesting resources held by other tasks in the set while holding resources requested by other tasks. In terms of data base management, the resources are data records or areas.

In many situations, deadlock can be avoided by limiting the update ability for a portion of the data base to a single task. This is not an attractive approach for distributed systems, since one of the motivating factors for a distributed DBMS is the sharing of data. If shared update is permitted in a DBMS, then the potential for deadlock exists. Deadlock may be handled by either detecting its existence and then resolving the conflict or by preventing all possible deadlock occurrences. Most existing single machine systems use deadlock detection and rollback as their means of treating the deadlock problem.

In a distributed DBMS, the most costly operation is intermachine communication. Therefore, the amount of intermachine communication becomes the dominant factor when evaluating the performance of deadlock detection and deadlock prevention schemes for distributed data bases. Deadlock prevention in a distributed DBMS requires that each task be informed as to which records that it may access are shared. For this information to be meaningful, only the records shared with currently active tasks should be included. This implies that whenever a task that updates shared records is activated, it must send a message to all other tasks. When this has taken place, the new task can proceed. Whenever a need to access a shared record arises, a prevention algorithm can be invoked to determine if the access may proceed. If not, the task is blocked until the resources are available.

Deadlock detection in a distributed DBMS involves first identifying [illegible] or more tasks blocking each other from a collection of shared

records. Once the set of conflicting resources are identified, one of the tasks must be rolled back to some point that will free the resources necessary to break the deadlock.

It is difficult to project the performance effects of deadlock detection and prevention in a distributed DBMS. A prevention scheme will block tasks in some situations in which no deadlock would actually occur. Detection takes no unnecessary actions in this respect. However, as indicated in the next subsection, rollback in a distributed DBMS could cause substantial performance degradation.

IV.2. ROLLBACK AND RECOVERY

Since all data base systems are susceptible to both hardware and software failure, recovery procedures are necessary to insure the integrity of the data base and to minimize the effect of system failures. Recovery procedures are similar to those for handling deadlock in that they become increasingly complex as the data base becomes more integrated. The major difficulty in data base recovery is that an erroneous application task may produce incorrect data which then may be accessed by other tasks. If not handled properly the bad data may have a cascading effect throughout the data base and thus damage its integrity as shown in Figure 5.

There are two extreme approaches to DBMS recovery for an application program failure. Some systems rollback only the terminated program while others rollback the entire data base to the point of initiation of the offending task. The former approach does not effect tasks which have no access to the polluted data although it may permit incorrect data to have been used by other application task. The latter approach is more conservative since it insures correct data at the cost of potentially rolling back tasks which did not interact with the polluted data.

Selective recovery is an intermediate recovery strategy which

entails rolling back only those tasks that are operating with polluted data. Overall system throughput and data base access would both increase under a selective recovery strategy. In order for a selective scheme to be worthwhile, data base integrity must be maintained. Therefore, the scheme must be certain of including all tasks that have used incorrect data in the recovery process. In order to accomplish selective rollback, information on the interaction between tasks must be maintained. The interaction information would take the form of a potential shared data list which can be computed from the sub-schemas of the application tasks prior to execution. Communication must be limited to one transmission to each back-end processor per rollback operation if performance is not to be effected. Maryanski and Fisher [2] have developed a selective recovery scheme for distributed data base systems.

IV.3. DATA CONVERSION

The conversion of data is one problem that is unique to distributed data base systems. The conversion problem has two aspects: the physical translation of the data and the logical conversion of the data base structure. The need for physical conversion arises when processors with different internal codes reside in the data base network. The code conversion problem is not difficult for any pair of machines. However, if distinct machines reside in the network, then it would be necessary for each processor node to contain K-1 translation routines. An alternative approach is to define a network standard code which is used in all intermachine communciation. This method results in a maximum of 2 translation routines per machine.

The most significant effort aimed at a generalized translation scheme for data bases is the University of Michigan Data Translation Project [3-5]. Under this approach, all data bases are described using a universal Stored Data Definition Language. The translations are driven

by tables produced by compilers for the SDDL and the Translation Definition Language (TDL). The TDL is employed to express the relationship between the source and target data bases. This translation methodology requires only one translation program at each DBMS node. However, the translator must be supplemented with a SDDL Table and (K-1) TDL Tables.

IV.4. OPERATING SYSTEMS

A DBMS on a single computer relies upon the operating system of the computer to perform functions such as I/O and task management. When the DBMS system software is distributed over a network its relationship with the operating systems of the network nodes is altered.

Several forms of operating systems may be applicable in a distributed DBMS. The most straightforward approach is to maintain a completely general purpose operating system on each processor. However, if primary memory is a critical resource on a processor, it may be desireable to employ a reduced version of the operating system. Since the DBMS software is distributed functionally between host and back-end processors, it may be desireable to distribute operating system functions in a similar manner. For instance, since the host machine does not perform any data base I/O operations, the appropriate drivers could be removed from that machine's operating system. The concept of subsetting general purpose operating systems can be extended to developing of a specially tailored operating system for data base machines [6]. Tailored operating systems are particularly well suited for hardware and stand alone data base machines [7-9].

IV.5. COMMUNICATIONS

If a distributed DBMS is to provide acceptable performance, inter-machine communication must be carried out efficiently. In addition, the communication process should be transparent to the application programmer.

These requirements can be satisfied by a standardized, portable communications interface that completely relieves the DBMS of any communication functions. Portability is an important factor if the network is to expand. It is highly desireable to have a standard communication system that can easily be implemented upon a new architecture. If portability is a requirement, implementation of the communication (and data base) software should be in a common high order language.. Descriptions of communication interfaces for data base systems built upon networks can be found in References [9-10].

IV.6. INTEGRITY

The distribution of a data base complicates the chore of maintaining the data base integrity. However, under certain circumstances, a distributed DBMS provides greater security than a single machine system. In a geographically distributed DBMS the involvement of additional computation and communication components increases the probability of a hardware failure. On the other hand, the communications software provides a means of checking the status of lines and processors and thus identifying a failure more rapidly than in the case of a single processor DBMS.

A DBMS with a dedicated back-end processor can be made more secure than a DBMS on a single, general purpose machine. The reason for increased security is that no unauthorized programs can be executed on the back-end machine since it is limited to the execution of data base commands. If the communications lines can be secured (not an easy task). Then the monitoring of data base activity can be prevented.

IV.7. REDUNDANCY

In a well designed, integrated data base, data redundancy should be quite rare. However, if the data base is to be distributed geographically,

it may become desireable from a performance standpoint to maintain redundant copies of data at various locations. Naturally, the update frequency of redundant data should be quite low. However, the updating of redundant data in a distributed DBMS produces considerable overhead since multiple data base operations must be generated.

Redundant data in a distributed DBMS becomes most troublesome when a recovery operation involves modified redundant data. The tasks of determining the effect on other programs and data of multiple copies of incorrect data items can become quite complex in even a moderately integrated distributed DBMS. Since the update of redundant data can not be performed at precisely the same moment at all nodes of the network, sequencing of the updates is also a problem for the recovery procedure.

At the present time, a recovery procedure for a distributed DBMS which can handle redundant data has not yet been proposed.

V. CONCLUSION

Because of increasingly complex systems and supporting software of minicomputers and their relatively low cost, much attention has been given to minicomputer networks or minicomputers in support of larger systems. With this interest it is important to consider that achieving data distribution over heterogeneous machines and systems is not a simple task. However, several developments are underway at multiple locations to provide such a facility which is independent of the hardware.

The considerations in this paper are alternatives and problems which we have faced in designed and implimenting a distributed system. Thus, the purpose of this paper is to enumerate and identify problems which must be solved when designing a network of machines and the attendant software for implementing a distributed data base management system.

REFERENCES

1. Maryanski, F.J. and Wallentine, V.E., "A Simulation Model of a Back-End Data Base Management System," Proc. Pittsburgh Conference on Modeling and Simulation, Apr. 1976, pp. 243-248.

2. Maryanski, F.J. and Fisher, P.S., "Rollback and Recovery in Distributed Data Base Systems," TR CS77-05 , Computer Science Department, Kansas State University, Manhattan, KS. 66506, Feb. 1977.

3. Yamaguchi, K. and Merten, A.G., "Methodology for Transferring Programs and Data," Proc. ACM SIGMOD Workshop, May 1974, pp. 141-155.

4. Merten, A.G. and Fry, J.P., "A Data Description Language Approach to File Translation," Proc. ACM SIGMOD Workshop, May 1974, pp. 191-205.

5. Birss, E. W. and Fry, J.P., "Generalized Software for Translating Data," Proc. AFIPS National Computer Conference, Vol. 45, June 1976, pp. 889-897.

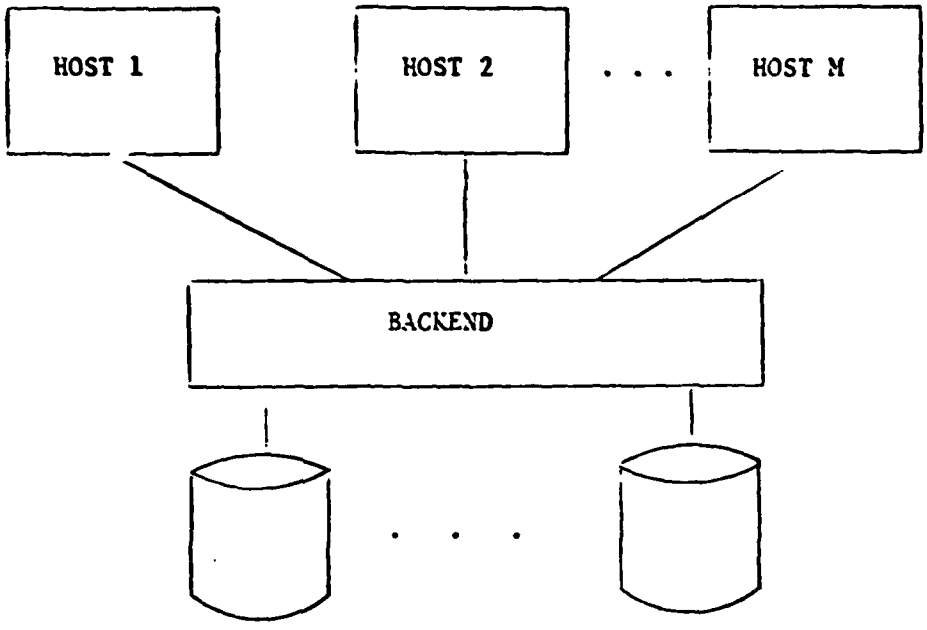
6. Baum, R.I. and Hsiao, D.K., "Database Computers--A Step Towards Database Utilities," IEEE Trans. on Computers, Vol. C-25, No. 12, Dec. 1976, pp. 1254-1259.

7. Lowenthal, E.I., "The Backend Computer," MRI Systems Corp., P.O. Box 9968, Austin, TX. 78766, Apr. 1976.

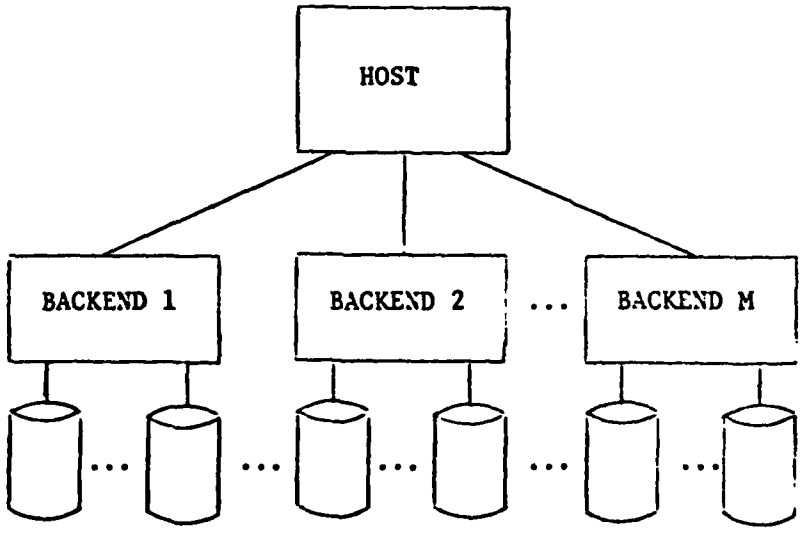
8. Ozkarahan, E.A., Schuster, S.A., and Smith, K.C., "RAP--An Associative Processor for Data Base Management, Proc. AFIPS National Computer Conference, Vol. 44, May 1975, pp. 379-387.

9. Marill, T. and Stern, D., "The Datacomputer--A Network Data Utility," Proc. AFIPS National Computer Conference, Vol. 44, May 1975, pp. 389-395.

10. Wallentine, V.E., "MIMICS--The Capabilities to be Developed," Computer Science Department, Kansas State University, Manhattan, KS. 66506, May, 1976.

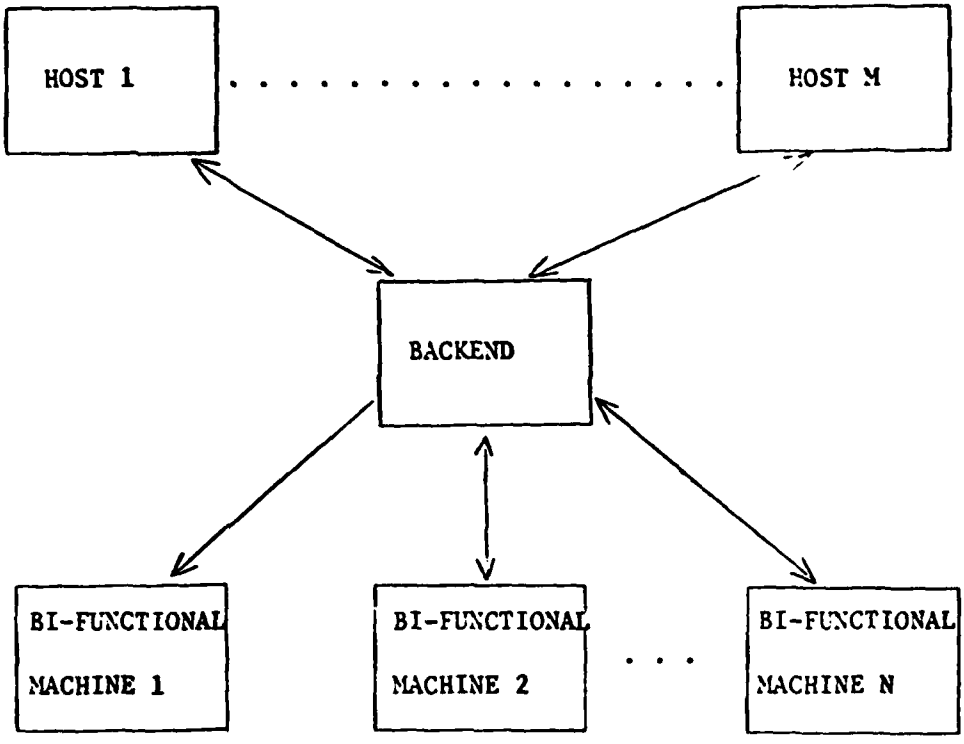


a. A Multi-host System with a Common Backend Machine Supporting DBMS Function



b. A Single Host System with Multiple Backend Systems Each Supporting a DBMS system

Figure 1: The Various Configurations Involving a Data Base Function



c. A Multi Host Backend and Bi-Functional (Host Backend Combination) Distributed Network.

Figure 1 (Continued): The Various Configurations Involving a Data Base Function

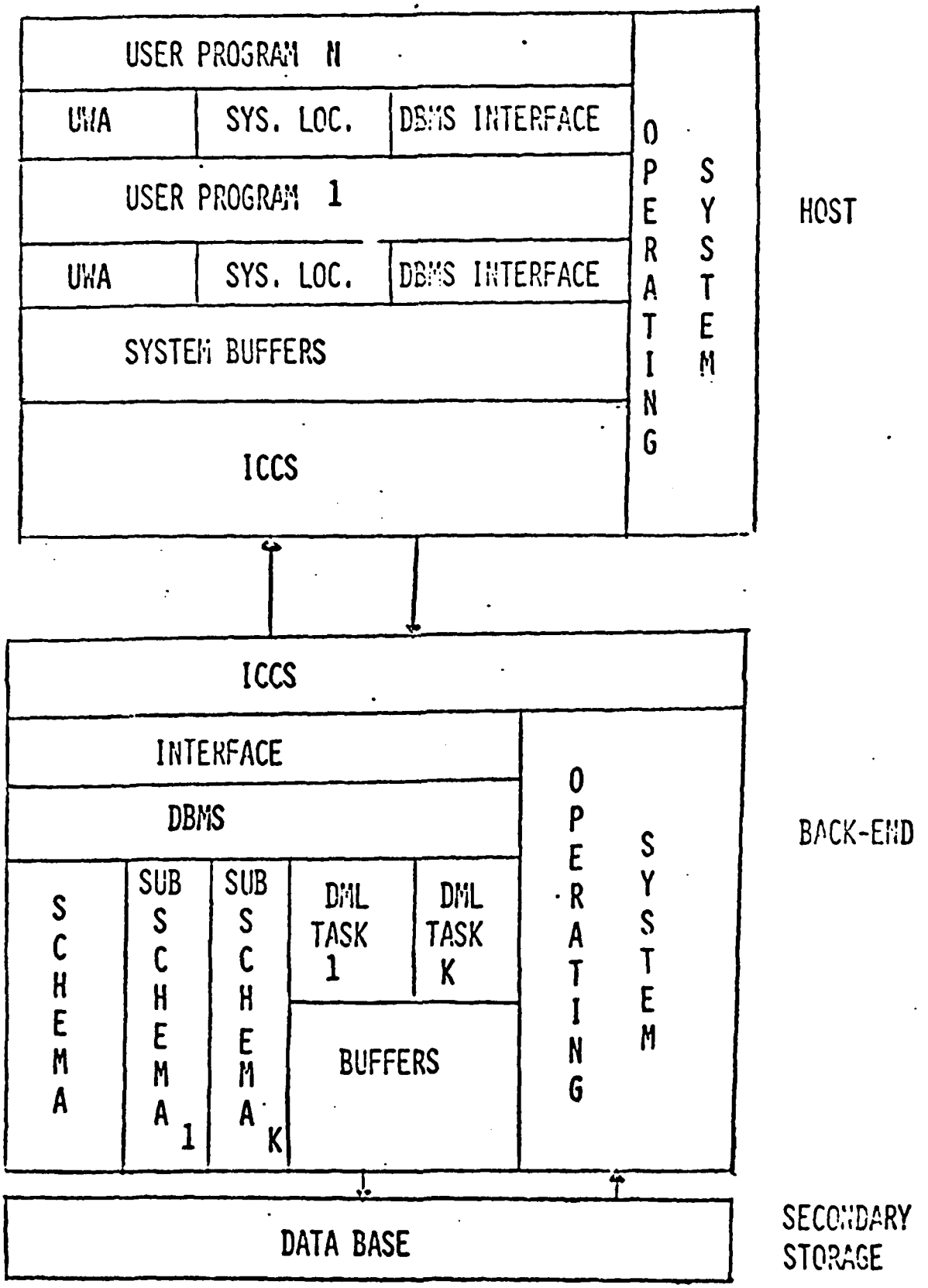


FIGURE 2: SOFTWARE DISTRIBUTION IN A HOST AND BACK-END SYSTEM

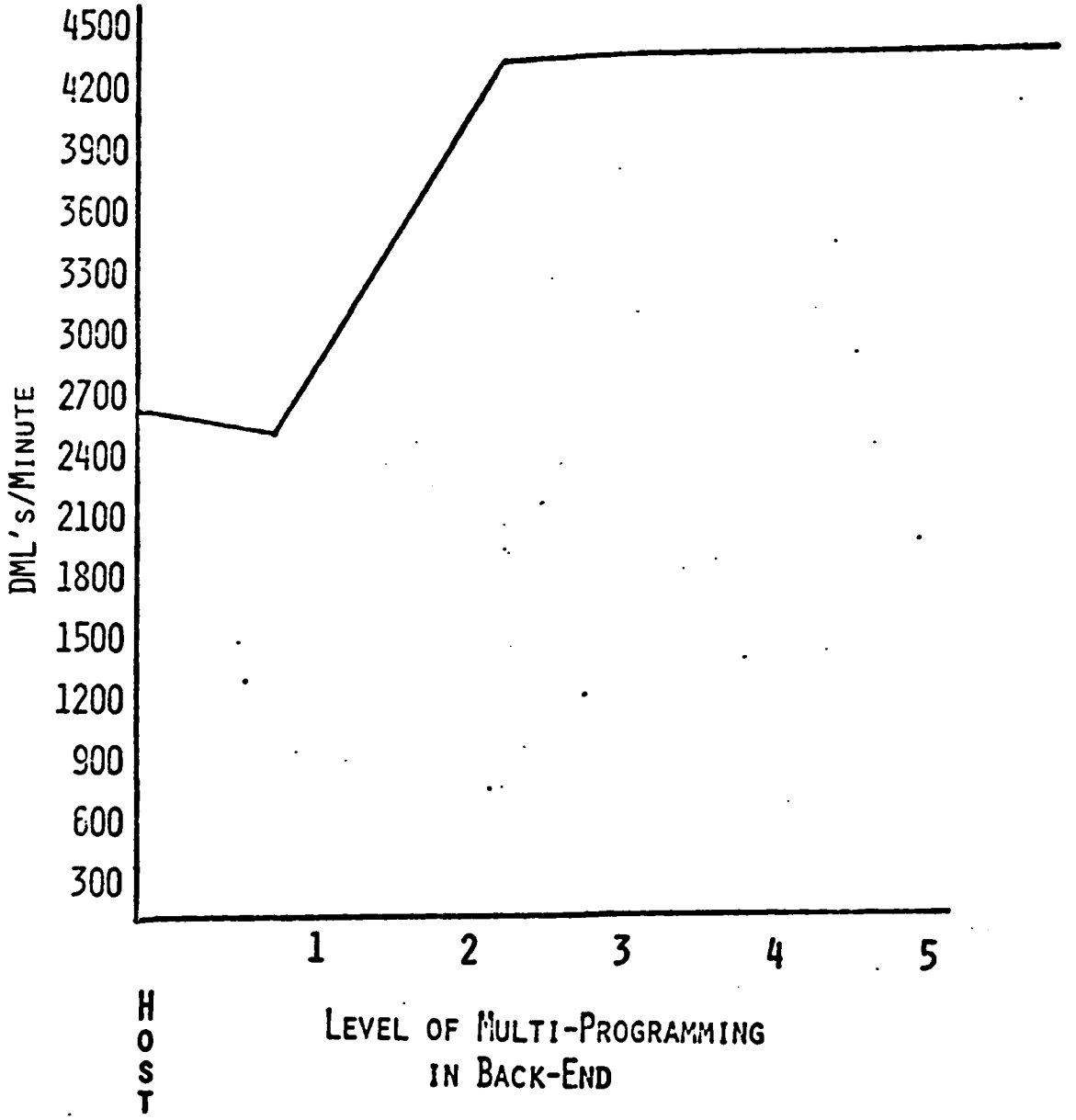


FIGURE 3: RELATIONSHIP OF DBMS REQUESTS AND A CONFIGURATION OF HOST ONLY AND HOST/BACK-END WHERE THE BACK-END IS MULTI-PROGRAMMED AT SEVERAL LEVELS

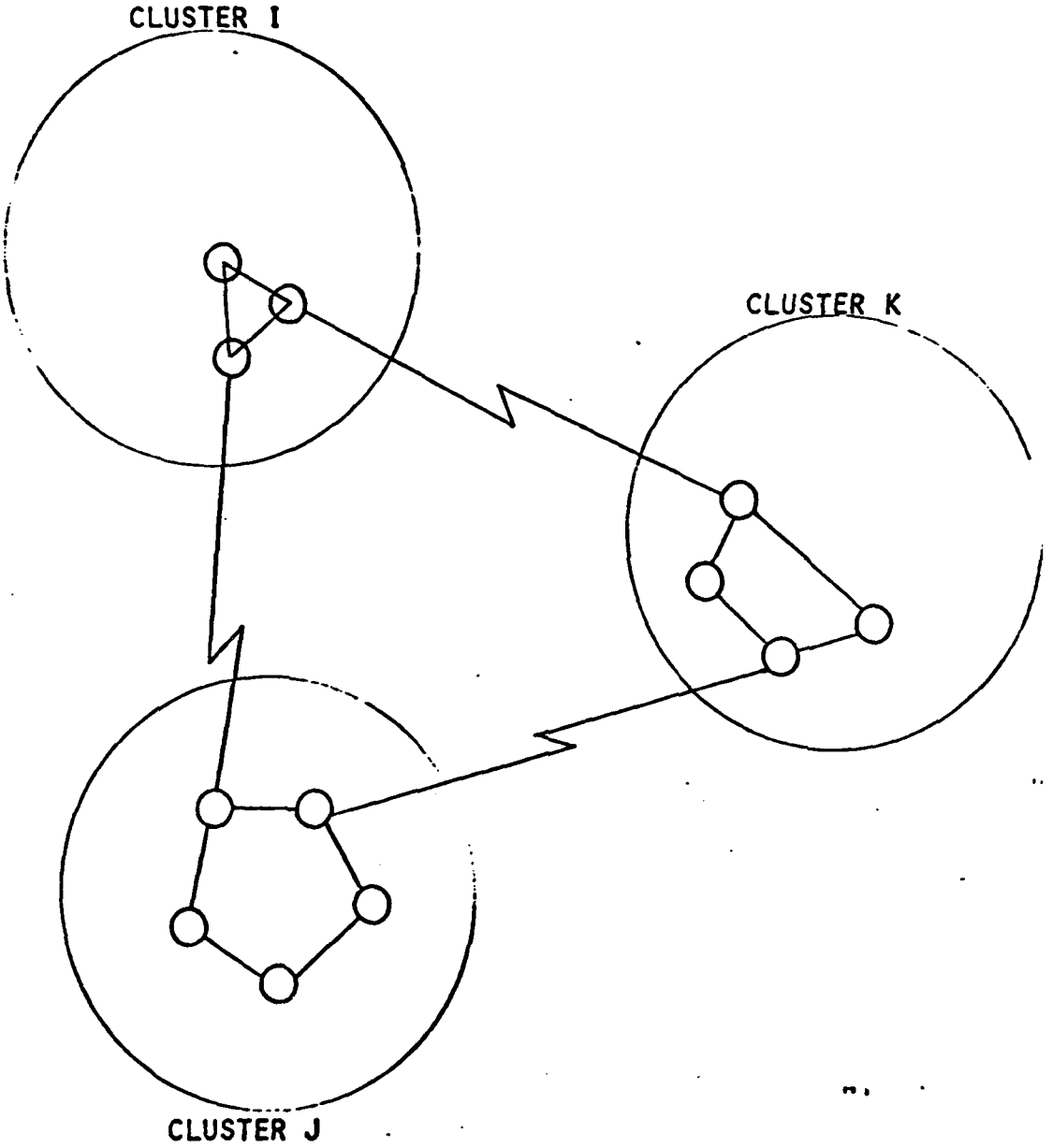


FIGURE 4: A NETWORK OF MACHINES WHERE EACH NODE OF THE NETWORK IS A CLUSTER OF ONE OR MORE MACHINES

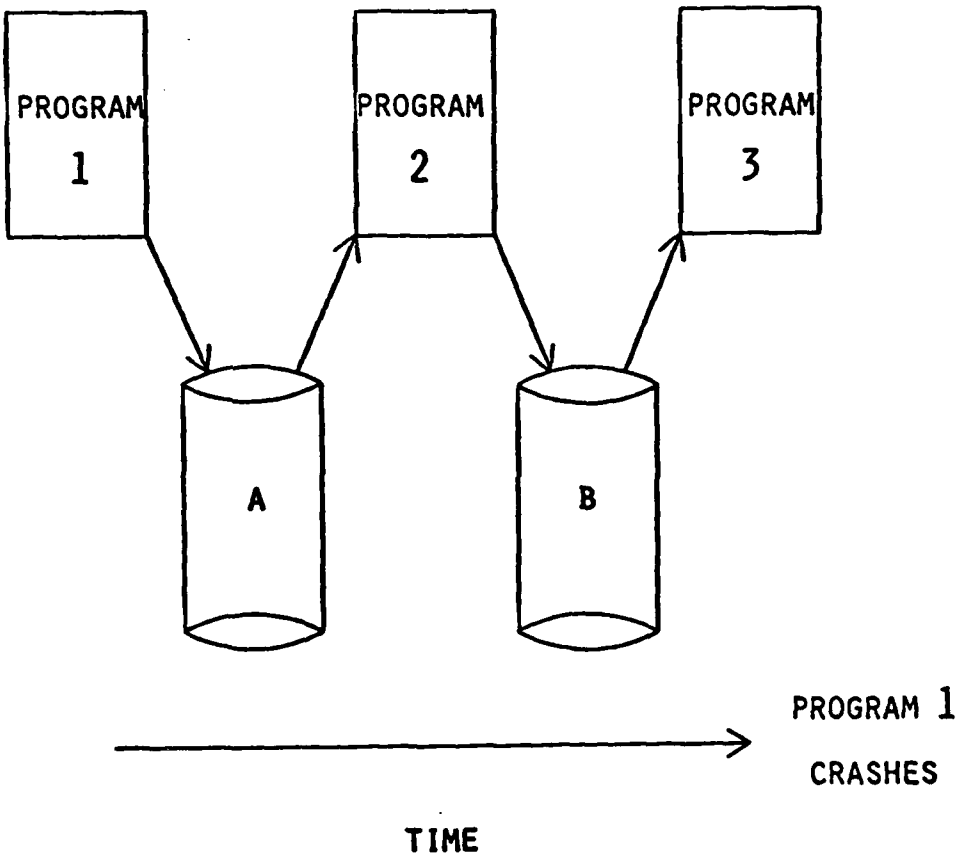


FIGURE 5: CASCADING EFFECT OF POLLUTED DATA